JOURNAL OF COMPUTER AND SYSTEM SCIENCES **52**, 506–521 (1996)
ARTICLE NO. 0038

SHINICHI MORISHITA

IBM Tokyo Research Laboratory, Tokyo, Japan

Received November 20, 1995

We study the application of the magic-sets transformation technique to Datalog$^{\neg}$ (function-free programs with negation) that may not have two-valued well-founded models. In this general setting, the well-founded model of the original program does not always agree with the well-founded model of the magic program derived by commonly used left-to-right sideways information-passing strategies on the query. In order to correct this disagreement we present a novel method that is obtained by slightly and naturally tailoring Van Gelder's alternating fixpoint technique to any magic program. © 1996 Academic Press, Inc.

1. INTRODUCTION

Much research has been done on the efficient evaluation of queries for deductive databases [29]. Of the various approaches investigated, the most promising is the magic-sets transformation [4–6, 21, 26, 29], because of its generality and efficiency. Basically, the magic-sets transformation simulates top-down evaluation with memoing and therefore generates the same set of queries, and the same set of answers to those queries, as top-down evaluation. Queries are expressed by auxiliary predicates called magic predicates, and each of the original rules is modified so that it fires only when values for these magic predicates are available, which significantly restricts the search space of non-magic facts to a subspace of those relevant to the query.

The magic-sets transformation was first developed for programs without negation (Horn logic programs). Programs without negation have a natural and well-accepted semantics defined by unique minimal Herbrand models. Programs with negation, however, may have many minimal Herbrand models. This problem has motivated many attempts to provide an intuitive semantics for programs with negation. The best-established of these attempts are the stable model semantics of Gelfond and Lifschitz [12], and the well-founded model semantics of Van Gelder *et al.* [30]. There is a close relationship between those two semantics: they are identical in many programs, and well founded total models are unique stable models. Some programs, however, may not have any stable models, whereas every program has a unique well-founded model. This robustness of the well-founded model semantics is accepted as a desirable property for programs with negation.

One of the main issues in deductive database research has been to extend the magic-sets transformation to programs with negation. Stratified programs constitute an important subclass of programs with negation, because the well-founded model of any stratified program is two-valued [30]; its positive portion can be computed efficiently in a bottom-up manner by iterating least fixpoint computation [1], and its negative portion can be obtained by complementing the positive facts. The problem in extending the magic-sets transformations to stratified programs is that a magic program resulting from transformation of a stratified program may not be stratified. Since the well-founded models of the original program and the magic program coincide on the query [14], constructive methods for computing well-founded models, such as Van Gelder's alternating fixpoint [31], can be used. However, these methods explicitly construct negative facts as well as positive facts. Several researchers [3, 2, 13, 25] have proposed methods that are cleverly designed to compute only positive facts.

If the well-founded model of a program is two-valued, we only need to generate positive facts. It would be useful to be able to determine whether the well-founded model of a given program is two-valued, but Papadimitriou and Yannakakis [18] have shown that this problem is undecidable. Ross [23] proposed a large class of programs, called modularly stratified programs, that have two-valued well-founded models and properly include all (locally) stratified programs. For modularly stratified function-free programs, Ross gave a bottom-up evaluation method that computes only positive facts, and showed that the method can be extended to magic programs. Ramakrishnan *et al.* [20] gave an improved technique called `Ordered_`

* This work was done during the author's visits to the Department of Computers Science at Stanford University and to IBM's Almaden Research Center while on leave from the IBM Tokyo Research Laboratory. A preliminary version of this paper appeared in the "Proceedings of the Twelfth ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, Washington DC, May 1993."

0022-0000/96 $18.00

$$p(X) \leftarrow t(X, Y, Z), \neg p(Y), \neg p(Z)$$
$$p(X) \leftarrow p_0(X)$$

FIG. 1. Datalog$^\neg$ program P.

`Search` for the magic programs derived from (left-to-right) modularly stratified function-free programs.

Methods specializing in handling modularly stratified programs, however, suffer from the problem that it is undecidable whether a given program is modularly stratified for all EDBs [24]. Put another way, they can be unsound if applied to non-modularly stratified programs, which may have three-valued well-founded models. When we consider a program whose well-founded model is three-valued, we need to construct negative as well as positive facts. Thus, restricting the search space via the magic-sets transformation becomes still more important. In this general setting, however, the well-founded model of the original program may not agree with that of the magic program derived by using left-to-right sideways information-passing (sip, for short) strategies on the given query [14]. This paper addresses that problem and offers a solution.

1.1. *Motivating Example*

Here we will present an example in which the well-founded model of the original program may not agree with that of the magic program on the given query. Consider the program P shown in Fig. 1. This program, cited from Ross [23], has t and p_0 as EDB predicates. Figure 2 shows a magic program for the query $\leftarrow p(a)$ derived by means of left-to-right sip strategies. Figure 3 shows an EDB, which is graphically represented in Fig. 4. $t(b_1, c_1, b_2)$, for instance, is represented by the edges from the node b_1 to the left node c_1 and the right node b_2. We attach an asterisk $(*)$ to the node c_2, because $p_0(c_2)$ is true.

P with the given EDB is not modularly stratified, because if we instantiate the first rule of P with $t(a, a, b_1)$, we see that $p(a)$ negatively depends on itself. Now observe that in the well-founded model of P, $p(c_2)$ is true, which makes $p(b_2)$ false. Further, since $p(c_1)$ is false, $p(b_1)$ is true, and hence $p(a)$ is false. In the well-founded model of MP, however, $p(a)$ is undefined because of new dependencies between ground atoms through magic facts introduced in the magic program MP. The troublesome obstacle is that any magic fact except $m\_p(a)$ is undefined, which prohibits us from inferring $p(b_1)$ and $p(c_2)$ that are true in the well-founded model of P.

$$m\_p(a)$$
$$m\_p(Y) \leftarrow m\_p(X), t(X, Y, Z)$$
$$m\_p(Z) \leftarrow m\_p(X), t(X, Y, Z), \neg p(Y)$$
$$p(X) \leftarrow m\_p(X), t(X, Y, Z), \neg p(Y), \neg p(Z)$$
$$p(X) \leftarrow m\_p(X), p_0(X)$$

FIG. 2. Magic program MP for $\leftarrow p(a)$.

$$p_0(c_2)\ t(a, a, b_1)\ t(b_1, c_1, b_2)\ t(b_2, c_2, b_3) \cdots t(b_n, c_n, c_{n+1})$$

FIG. 3. An EDB for P.

One can trivially solve this problem by adding all magic facts to the magic program, but the resulting magic program is equivalent to the original program, and hence this modification does not restrict the search space at all. We want to reduce the number of magic facts to be considered. In our example, if we use $m\_p(a)$, $m\_p(b_1)$, $m\_p(c_1)$, $m\_(b_2)$, and $m\_p(c_2)$, we can determine the falsity of query $p(a)$, even though we ignore any magic fact having any node under b_3 as its argument.

Kemp *et al.* proposed two methods [14, 15] of resolving the disagreement between the well-founded models of the original program and the magic program. In the first method [14], they propose using special sip strategies to derive a magic program that generates many more magic facts than one created via left-to-right sip strategies. As an example, they give well-founded sip strategies that may evaluate subgoals in parallel instead of from left to right. If applied to our example, the well-founded sip strategies in this method evaluate in parallel the two negated subgoals in the first rule of P, and the magic program derived via this sip strategy is the same as MP except that the third rule does not have $\neg p(Y)$. As a result, all magic facts are derived by the new magic program. The other method [15] allows any sip strategies to derive magic programs and considers all magic facts that are true or undefined in the well-founded model of the magic program. Note that, if applied to our example, this method also results in the use of all magic facts. Both methods thus employ all magic facts.

We follow the basic idea of the latter approach; that is, we allow any sip strategies and add some magic facts to the magic program. Kemp *et al.*'s method is static in the sense that it first computes the set of all magic facts that are true or undefined and then starts computing the well-founded model of the magic program with that set. Put another way, the method uses a fixed set of magic facts during the computation. On the other hand, we propose a technique that dynamically adds some magic facts to the magic program at each step of the computation.

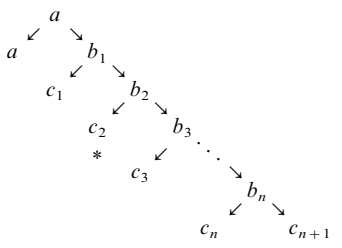

FIG. 4. A representation of the EDB in Fig. 3.

1.2. *New Method*

We modify Van Gelder's alternating fixpoint method [31], a standard method for computing well-founded models, for use with magic programs.

First, we briefly review Van Gelder's alternating fixpoint method. This generates a sequence of underestimates of true facts (subsets of all true facts) and a sequence of overestimates (sets of all true facts plus some others that are assumed to be true) by alternating between the following two phases:

- Perform forward reasoning until no more inferences are possible by assuming any negative literal whose atom does not appear in the previous *overestimate* (in the initial step we do not assume any negative literal). The new *underestimate* is the set of all facts inferred.
- Perform forward reasoning until no more inferences are possible by assuming any negative literal whose atom does not appear in the previous *underestimate*. The new *overestimate* is the set of all facts inferred.

The sequence of underestimates increases monotonically, the sequence of overestimates decreases monotonically, and both sequences finally reach fixpoints simultaneously. A fact is true in the well-founded model of the program if and only if it is in the final underestimate, and a fact is false if and only if it does not appear in the final overestimate.

We are now in a position to present our method. For magic programs, we change the first phase of Van Gelder's alternating fixpoint and compute an underestimate by using *any magic facts in the previous overestimate* as well as any negative literal whose atom does not appear in the previous overestimate. The intuition behind this modification comes from the observation that the set of magic facts in any overestimate is large enough to allow us to derive true non-magic facts relevant to the query. Although our method is a slight modification of Van Gelder's alternating fixpoint technique, it is different in spirit. Van Gelder's method uses only negative information from the underestimate computation in computing overestimates, and vice versa. Our method also uses limited positive information, namely magic facts, from the previous overestimate to control the subsequent underestimate.

We illustrate our method by applying it to our example:

1. The first underestimate is $\{m\_p(a)\}$. In what follows, although any underestimate and overestimate include all the EDB facts presented in Fig. 3, we omit them for simplicity.
2. The first overestimate is

$$\{p(a), p(b_1), ..., p(b_n), p(c_2), m\_p(a), m\_p(b_1), ..., m\_p(b_n), m\_p(c_1), ..., m\_p(c_{n+1})\}.$$

3. The second underestimate is

$$\{p(b_n), p(c_2), m\_p(a), m\_p(b_1), ..., m\_p(b_n), m\_p(c_1), ..., m\_p(c_{n+1})\}.$$

Since $m\_p(c_2)$ is in the previous overestimate, we can infer $p(c_2)$. We can also infer $p(b_n)$, because $m\_p(b_n)$ is in the previous overestimate, and $p(c_n)$ and $p(c_{n+1})$ do not appear in the previous overestimate. If the original alternating fixpoint had been applied, since no magic facts in the previous overestimate would have been derived, we would have had $\{m\_p(a)\}$ as the second underestimate.

4. The second overestimate is

$$\{p(a), p(b_1), p(c_2), m\_p(a), m\_p(b_1), m\_p(c_1), m\_p(b_2), m\_p(c_2)\}.$$

Since $p(c_2)$ is true in the previous (second) underestimate, our method generates no magic fact that has a node under b_3 as its argument.

5. The third underestimate is

$$\{p(b_1), p(c_2), m\_p(a), m\_p(b_1), m\_p(c_1), m\_p(b_2), m\_p(c_2)\}.$$

6. The third overestimate is

$$\{p(b_1), p(c_2), m\_p(a), m\_p(b_1), m\_p(c_1), m\_(b_2), m\_p(c_2)\}.$$

7. The fourth underestimate is equal to the third one.
8. The fourth overestimate is equal to the third one.

If applied to our example, both of Kemp *et al.*'s methods [15] add all magic facts to the magic program, and therefore require n steps to terminate. Thus in this case our method generates significantly fewer magic facts than Kemp *et al.*'s. Generally, our technique improves on Kemp *et al.*'s in the sense that it always generates fewer magic facts in the final step of the computation.

Our method of computing an answer to the query in the well-founded model of the original program is sound and complete. Let Q be a ground instance of the query. The truth value of Q in the well-founded model of the original program can be determined as follows: Q is true if and only if Q is in the final underestimate, and Q is false if and only if Q does not appear in the final overestimate. In our example, $p(a)$ does not appear in the final overestimate, and hence we can conclude that $p(a)$ is false. We will also prove that the sequence of overestimates decreases monotonically. Thus if the original program is function-free, the sequence reaches a fixpoint in a finite number of steps. The sequence of underestimates, however, does not always increase monotonically.

For instance, in our example the third underestimate does not include the second one. This non-monotonic nature makes the correctness of our method non-trivial.

2. PRELIMINARIES

In this section we review the definitions of Datalog, well-founded models, and the magic-sets transformation. We basically follow the notations and terminologies of deductive databases given in Beeri and Ramakrishnan [6], Ullman [29], and Van Gelder *et al.* [30].

2.1. *Datalog and Datalog*$^{\neg}$

DEFINITION 2.1. An *atom* has the form $p(t_1, ..., t_k)$, where p is a *predicate symbol* (or just *predicate*) and each t_i is a term. A *ground term* is a variable-free term. A *ground atom* (or *fact*) is a variable-free atom. A *literal* is an atom or a negated atom of the form $\neg A$, where A is an atom. An atom is called a *positive* literal, and a negated atom is called a *negative* literal. Let L be a literal. If L is a positive literal A, the *complement* of L is $\neg A$. If L is a negative literal $\neg A$, the *complement* of L is A. The complement of L is denoted by $\neg \cdot L$.

A *rule* has the form $A \leftarrow L_1, ..., L_n$, where A is a positive literal and each L_i is a literal. In the rule A, which appears to the left of $\leftarrow$, is called the *head* $L_1, ..., L_n$, which appears to the right of $\leftarrow$, is called the *body*. Each L_i is called a *subgoal*. We assume that rules satisfy *range restrictedness*; that is, each variable in the head of the rule also appears in its body. A ground instance of a rule is obtained by substituting ground terms for variables in the rule. A *program* is a set of rules. A *function-free* program is a program that does not have any function symbol. We will call a function-free program without negation a *Datalog* program. We also call a function-free program that may have negation a *Datalog*$^{\neg}$ program.

Let P be a program. H_P denotes the Herbrand base of P; that is, the set of all ground atoms (facts) made from constants and symbols appearing in P. Let I be a set of literals. Then, $\neg \cdot I$ is the set in which each literal in I has been complemented. Let I^+ be the set of all positive literals in I, and let I^- be the set of all negative literals in I. If J is a subset of H_P, $\bar{J}$ denotes $\neg \cdot (H_P - J)$.

EXAMPLE 2.1. Let I be $\{p(a), \neg p(b), p(c), \neg p(d), p(e)\}$. $I^+ = \{p(a), p(c), p(e)\}$, and $I^- = \{\neg p(b), \neg p(d)\}$. Let H_P be $\{p(a), p(b), p(c), p(d), p(e), p(f), p(g)\}$, and let J be $\{p(a), p(c), p(e)\}$, which is a subset of H_P. Then,

$$\bar{J} = \neg \cdot (H_P - J) = \{\neg p(b), \neg p(d), \neg p(f), \neg p(g)\}.$$

PROPOSITION 2.1. *Let I, J, and K be subsets of H_P. Then,*

$$I \cap J \subseteq K \quad \text{if and only if} \quad \neg \cdot J \cap \bar{K} \subseteq \bar{I}.$$

Proof. Observe that $I \cap J \subseteq K$ if and only if $J \cap (H_P - K) \subseteq (H_P - I)$. ∎

We also need basic lattice-theoretic terminologies. We assume that the reader is familiar with the concept of ordinals [16].

DEFINITION 2.2. Let S be a set and let 2^S be the power-set of S. Let T be a mapping from 2^S to itself; that is, let $T: 2^S \to 2^S$. The ordinal power of T are inductively defined as follows:

$$T \uparrow 0 = \phi$$
$$T \uparrow \alpha = T(T \uparrow \alpha - 1); \qquad \alpha \text{ is a successor ordinal}$$
$$T \uparrow \alpha = \bigcup \{T \uparrow \beta \mid \beta < \alpha\}; \qquad \alpha \text{ is a limit ordinal.}$$

Ordinals are denoted by the Greek letters, $\alpha, \beta, \gamma, ...$. The first limit ordinal is denoted by ω as usual. A subset I of S is called a *fixpoint* of T when $T(I) = I$. The least fixpoint of T, denoted by $\mathrm{lfp}(T)$, is a subset of any fixpoint of T and is also a fixpoint of T.

DEFINITION 2.3. Let S_1 and S_2 be sets, and let $T: 2^{S_1} \to 2^{S_2}$ be a mapping. T is called *monotonic* if for all subsets I and J of S_1 such that $I \subseteq J$, $T(I) \subseteq T(J)$. Let $\mathbf{N}$ denote the set of natural numbers. T is called *continuous* if for all subsets I_i $(i \in \mathbf{N})$ of S_1 such that $I_i \subseteq I_{i+1}$, $T(\bigcup \{I_i \mid i \in \mathbf{N}\}) = \bigcup \{T(I_i) \mid i \in \mathbf{N}\}$.

Note that if T is continuous, then T is monotonic. The following well-known properties attributed to Knaster, Tarski, and Kleene are frequently used throughout this paper:

THEOREM 2.1. *Let T be a mapping from 2^S to 2^S, where S is an arbitrary set. Then we have*:

- *If T is monotonic, the least fixpoint of T exists. Furthermore, we have the following equalities*:

$$\mathrm{lfp}(T) = \bigcap \{I \subseteq S \mid T(I) \subseteq I\} = T \uparrow \tau \qquad \text{for some ordinal } \tau.$$

- *If T is continuous, the least fixpoint of T is equal to $T \uparrow \omega$.*

2.2. *Well-Founded Partial Models*

We review the definition of well-founded partial models presented in Van Gelder *et al.* [30].

DEFINITION 2.4. Let S be a set of ground literals. If a literal L appears in $\neg \cdot S$, then L is called *inconsistent* with S. Otherwise, L is *consistent* with S. For any ground atom A, if S does not include both A and $\neg A$, then S is called *consistent*.

Let P be a program. A *partial interpretation* I is a consistent set of ground literals whose atoms are in H_P. A *total*

interpretation is a partial interpretation that contains every ground atom of H_P or its negation. A ground literal is *true* in I when it appears in I, *false* in I when its complement is in I, and *undefined* otherwise. A ground instance of a rule in P is true in I when the head is true in I or some subgoal is false in I; it is false when the head is false and all subgoals are true. A *total model* of P is a total interpretation such that every ground instance of any rule in P is true in the model. A *partial model* of P is a partial interpretation that can be extended to a total model.

DEFINITION 2.5. Let P be a program that may have negative subgoals in its rules. $\mathbf{T}_P: 2^{H_P \cup \neg \cdot H_P} \to 2^{H_P}$ is a mapping from the powerset of ground literals to the powerset of ground positive literals, defined as follows: for any set of ground literals I, H is in $\mathbf{T}_P(I)$ if and only if H is the head of a ground instance of a rule in P, all positive subgoals are in I^+, and all negative subgoals are in I^-. Intuitively, $\mathbf{T}_P(I)$ is the set of all facts immediately derived from I by using rules in P.

Let N denote a set of ground negative literals. Let $\mathbf{T}_P[N]: 2^{H_P} \to 2^{H_P}$ be a mapping from the powerset of ground positive literals to itself whose definition is as follows; for any set of ground positive literals J, H is in $\mathbf{T}_P[N](J)$ if and only if H is the head of a ground instance of a rule in P such that all positive subgoals are in J and all negative subgoals are in N. Put another way, $\mathbf{T}_P[N](I)$ is the set of all facts immediately derived from I and N by using rules in P.

Let $\mathbf{S}_P(N)$ denote $\mathbf{T}_P[N] \uparrow \omega$. $\mathbf{S}_P$ is a mapping from the powerset of ground negative literals to the powerset of ground positive literals. $\mathbf{S}_P(N)$ is the set of all facts inferred by performing forward reasoning until no more inferences are possible by assuming all negative literals in N.

We can easily show the following properties.

THEOREM 2.2. *$\mathbf{T}_P$ and $\mathbf{T}_P[N]$ are continuous, and $\mathbf{S}_P$ is monotonic.*

PROPOSITION 2.2. *Let P be a program, let N be a set of ground negative literals, and let X be an arbitrary subset of $\mathbf{S}_P(N)$. Then, $\mathbf{S}_{P \cup X}(N) = \mathbf{S}_P(N)$.*

DEFINITION 2.6. Let P be a program and let I be a partial interpretation of P. We say that $U \subseteq H_P$ is an *unfounded set* of P with respect to I if each $A \in U$ satisfies the requirement that for each ground rule R of P whose head is A, one of the following conditions holds:

1. Some (positive or negative) subgoal is false in I; that is, it is in $\neg \cdot I$.
2. Some positive subgoal occurs in U.

A literal that makes one of the above conditions true is called a *witness of unusability* for rule R (with respect to I). The union of all unfounded sets with respect to I is also unfounded and is called the *greatest unfounded set*, which we will denote by $\mathbf{U}_P(I)$. Then, $\mathbf{W}_P(I)$ is the mapping defined by

$$\mathbf{W}_P(I) = \mathbf{T}_P(I) \cup \neg \cdots \mathbf{U}_P(I).$$

Note that $\mathbf{U}_P$ and $\mathbf{W}_P$ are monotonic mappings, and hence the least fixpoint of $\mathbf{W}_P$ exists. The following property holds [30]:

THEOREM 2.3. *For each ordinal α, $\mathbf{W}_P \uparrow \alpha$ is a partial (or total) model of P.*

According to the above property, the least fixpoint of $\mathbf{W}_P$ is a partial (or total) model of P and is called the *well-founded (partial) model* of P. The following property [30] allows us to compute $\mathbf{U}_P(\mathbf{W}_P \uparrow \alpha)$ constructively from $\mathbf{W}_P \uparrow \alpha$. Recall that $\overline{(\mathbf{W}_P \uparrow \alpha)^+} = \neg \cdot (H_P - (\mathbf{W}_P \uparrow \alpha)^+)$.

THEOREM 2.4. $\mathbf{U}_P(\mathbf{W}_P \uparrow \alpha) = H_P - \mathbf{S}_P(\overline{(\mathbf{W}_P \uparrow \alpha)^+})$.

Van Gelder [31] proposed an elegant method for computing the well-founded model of a given program in a constructive manner:

THEOREM 2.5. *Let P be a program, and let α be an ordinal. Let $\mathcal{U}_\alpha$ and $\mathcal{O}_\alpha$ be sets of positive facts that are computed as follows*:

$$\mathcal{U}_0 = \mathbf{S}_P(\phi) \qquad \mathcal{O}_0 = \mathbf{S}_P(\overline{\mathcal{U}_0})$$
$$\mathcal{U}_{\alpha+1} = \mathbf{S}_P(\overline{\mathcal{O}_\alpha}) \qquad \mathcal{O}_{\alpha+1} = \mathbf{S}_P(\overline{\mathcal{U}_{\alpha+1}}).$$

If α is a limit ordinal,

$$\mathcal{U}_\alpha = \bigcup_{\beta < \alpha} \mathcal{U}_\beta \qquad \mathcal{O}_\alpha = \mathbf{S}_P(\overline{\mathcal{U}_\alpha}).$$

Then, $\mathcal{U}_\alpha$ is a subset of all the true facts in the well-founded model of P. The sequence of $\mathcal{U}_\alpha$ increases monotonically. $\mathcal{O}_\alpha$ is a superset of all the true facts and all the undefined facts. The sequence of $\mathcal{O}_\alpha$ decreases monotonically. $\mathcal{U}_\alpha \subseteq \mathcal{O}_\beta$ for any pair of α and β. Both sequences finally reach fixpoints simultaneously. Let τ be an ordinal such that $\mathcal{U}_\tau = \mathcal{U}_{\tau+1} = \cdots$ and $\mathcal{O}_\tau = \mathcal{O}_{\tau+1} = \cdots$. Then, $\mathcal{U}_\tau \cup \overline{\mathcal{O}_\tau}$ is equal to the well-founded model of P. $\mathcal{U}_\tau$ is the set of all true facts, and $\mathcal{O}_\tau$ is the set of all true facts and all undefined facts.

DEFINITION 2.7. $\mathcal{U}_\alpha$ is called an *underestimate* (of all the true facts), and $\mathcal{O}_\alpha$ is called an *overestimate*. $\mathcal{U}_\tau \cup \overline{\mathcal{O}_\tau}$ is called the *alternating fixpoint*.

2.3. *Magic-Sets Transformation*

In this section we study the application of the magic-sets transformation [6], which was originally developed for Datalog, to Datalog$^\neg$. When dealing with Datalog$^\neg$ we temporarily ignore negation symbols and pass binding information into negated subgoals. First, we review the definition of sideways information-passing strategies [6], which defines how to pass values from one atom to another in the rules of the program.

DEFINITION 2.8. Let R be a rule whose head is called with some arguments bound to constants, and let H_b be the special atom denoting the head projected on its bound arguments. Let $Literals(R)$ denote the set of all literals in the body of R. A *sideways information-passing* strategy for rule R is a labeled graph that meets the following conditions:

- Each node is either a subset or a member of $Literals(R) \cup \{H_b\}$.
- Each arc is of the form $N \rightarrow_\chi L$ with label χ, where N is a subset of $Literals(R) \cup \{H_b\}$, L is a member of $Literals(R)$, and χ is a set of variables such that each variable in χ appears both in L and in a member of N.
- There exists a total ordering of the literals in $Literals(R) \cup \{H_b\}$ in which H_b is the first, such that for each arc all literals at its tail precede the literal at its head, and such that the literals that do not appear in the sip strategy follow all others.

This definition of sip strategies, which is cited from Beeri and Ramakrishnan [6], models a sip strategy as a graph. Although no graph properties are used in the definition, a graphical representation can help us to understand complicated sip strategies. The above definition includes as many strategies as possible. For instance, it allows us to express a strategy that evaluates subgoals in the body in parallel. We may have a sip strategy in which multiple arcs enter or leave a node. In practice, however, the most common way of evaluating a rule is to proceed from left to right, carry along all the variables that have been computed so far, and pass all available binding information to subgoals. Such sip strategies, usually called *left-to-right*, are formally defined as follows:

DEFINITION 2.9. A sip strategy for rule R is *left-to-right* if, for each arc in a sip strategy of the form $N \rightarrow_\chi L$, N is the union of $\{H_b\}$ and the set of all literals that appear to the left of L in the body of R and χ is the set of all variables that appear both in L and in a member of N. Note that in any left-to-right sip strategies, any literal has only one entering arc.

EXAMPLE 2.2. Consider the rule

$$p(X) \leftarrow t(X, Y, Z), \neg p(Y), \neg p(Z).$$

Suppose that the argument of the head of the rule is bound. The left-to-right strategy is represented by the following sip strategy:

$$\begin{aligned}(s1)\quad & \{p(X)\} \rightarrow_{\{X\}} t(X, Y, Z)\\ & \{p(X), t(X, Y, Z)\} \rightarrow_{\{Y\}} \neg p(Y)\\ & \{p(X), t(X, Y, Z), \neg p(Y)\} \rightarrow_{\{Z\}} \neg p(Z).\end{aligned}$$

Another strategy is to pass the binding information in the head to $t(X, Y, Z)$ first and then evaluate $\neg p(Y)$ and $\neg p(Z)$ in parallel. This strategy can be illustrated by the following sip strategy:

$$\begin{aligned}(s2)\quad & \{p(X)\} \rightarrow_{\{X\}} t(X, Y, Z)\\ & \{p(X), t(X, Y, Z)\} \rightarrow_{\{Y\}} \neg p(Y)\\ & \{p(X), t(X, Y, Z)\} \rightarrow_{\{Z\}} \neg p(Z).\end{aligned}$$

Let us now focus on how binding information is passed to $\neg p(Z)$. The first sip strategy (s1) passes all the available information to $\neg p(Z)$; that is, it uses the bindings for all variables that have been computed before it considers $\neg p(Z)$. On the other hand, the second sip strategy (s2) does not consider the bindings of the variable in $\neg p(Y)$. Since (s1) uses the bindings of Y in $\neg p(Y)$, (s1) passes to $\neg p(z)$ no more bindings than (s2) does. In this sense, (s1) is more efficient than (s2). The reader might feel that there is no advantage in using (s2). We will reconsider this problem after defining the magic-sets transformation.

Next, we define the binding pattern of predicates in rules according to the sip strategy chosen for each rule.

DEFINITION 2.10. A binding pattern, or *adornment*, for an n-ary predicate symbol is represented as a string a of length n on the alphabet $\{b, f\}$, where b stands for *bound* and f stands for *free*. Let P be a Datalog$^\neg$ program, and let Q be a query against P. We construct a new, adorned version of P, denoted by P^{ad}, as follows.

We will generate a set of adorned predicates and mark each element one by one. First we start from Q and generate the adorned version of the predicate of Q, in which the positions bound in Q are designated as bound. The adorned version of Q is unmarked at this point. While there exists an unmarked element p^a, mark p^a as processed and perform the following steps for each rule R that has p in its head:

1. Choose a sip strategy for R whose head is called with the binding pattern a.
2. Replace each predicate p_i in the body of R with the adorned occurrence $p_i^{a_i}$, where an argument of p_i is bound in a_i only if the argument is a constant or the variable in that argument appears in the label of an arc in the chosen sip strategy coming into the literal of p_i. Unless $p_i^{a_i}$ has already been marked, it is unmarked and therefore is considered later. The arguments of the predicate in the adorned rule are the same as those in the original one.
3. Add the resultant adorned rule to P^{ad}.

EXAMPLE 2.3. Let us continue Example 2.2. Suppose that a query $\leftarrow p(c)$ is given. The adorned version of p is p^b. Then, the following is the only rule that has p in its head:

$$p(X) \leftarrow t(X, Y, Z), \neg p(Y), \neg p(Z).$$

If we choose (s1) or (s2) as the sip strategies, the adorned version of the above rule is

$$p^b(X) \leftarrow t^{bff}(X, Y, Z), \neg p^b(Y), \neg p^b(Z).$$

When P is a Datalog program, P with Q and P^{ad} with the adorned version of Q are equivalent with respect to the query [6]; that is, for any assignment of constants to the variables of Q and its adorned version, the two programs produce the same answer for the resulting queries on Q and its adorned version. It is easy to check that Datalog$^\neg$ programs also have this property; that is, the well-founded models of P and P^{ad} agree on Q. Thus, in what follows, we consider only P^{ad} with the adorned version of Q, and not P with Q.

We are now in a position to introduce the magic-sets transformation that creates auxiliary predicates called *magic predicates* for original predicates and modifies the adorned program so that rules fire only when values for these magic predicates are available.

DEFINITION 2.11. The *magic predicate* for an adorned predicate p^a is denoted by m_p^a. The arity of the magic predicate is the number of occurrences of b in the adornment a of p. Let A be an atom, that is, a positive literal. The *magic atom* for A, which we will denote $m[A]$, is an atom whose predicate is the magic predicate for the predicate of A and whose arguments correspond to the bound arguments of A, respectively. Let L be a negative literal that has the form $\neg A$. The magic atom for L, which will be denoted by $m[L]$, is the magic atom $m[A]$ for A.

EXAMPLE 2.4. The magic predicates for p^b and t^{bff} are m_p^b and m_t^{bff} respectively. The magic atoms for $p^b(X)$, $\neg p^b(Y)$, and $t^{bff}(X, Y, Z)$ are $m\_p^b(X)$, $m\_p^b(Y)$, and $m\_t^{bff}(X)$, respectively. Thus

$$m[p^b(X)] = m\_p^b(X), m[\neg p^b(Y)] = m\_p^b(Y),$$

and

$$m[t^{bff}(X, Y, Z)] = m\_t^{bff}(X).$$

We will distinguish occurrences of the same predicate with different adornments. Therefore, the magic atom for any non-magic ground atom is unique.

DEFINITION 2.12. Let P be a Datalog$^\neg$ program, and let P^{ad} be an adorned version of P with respect to query Q. The magic-sets transformation produces a *magic program* MP for P^{ad} as follows:

1. For each rule R in P^{ad}, and for each literal L in its body, we generate a *magic rule* defining the magic atom $m[L]$ for L in the following manner, and add the magic rule to MP.

   • Let $N \rightarrow_\chi L$ be the only arc entering L in the sip strategy. Then, the head of the magic rule is $m[L]$, and the body contains the magic atom for the head of R if N contains the special atom for the head, and all literals both in N and in the body of R.

   • If there is more than one arc entering L, we define the magic rule defining $m[L]$ in two steps. First, for each arch $N_j \rightarrow_{\chi_j} L$, we define a rule such that the head is $label_j(\chi_j)$ and the body is the same as the body of the magic rule in the case where there is a single arc entering L. Then, the body of the magic rule defining $m[L]$ is the conjunction of $label_j(\chi_j)$ for all arcs entering L.[1]

2. Each rule in P^{ad} is modified by the addition of the magic atom for the head to its body. We add the resultant rule to MP.

3. We create a seed for the magic predicates. The seed is the magic atom for the adorned version of Q and is added to MP.

Although we do not give the definition of generalized supplementary magic-sets transformation [6, 29], all the results presented in this paper can be extended straightforwardly to the generalized transformation.

EXAMPLE 2.5. Consider the query $\leftarrow p(a)$ and the rule $p(X) \leftarrow t(X, Y, Z), \neg p(Y), \neg p(Z)$. If we choose the sip strategy (s1) given in Example 2.2, the magic-sets transformation generates the following magic program:

$$m\_p^b(a)$$

$$m\_t^{bff}(X) \leftarrow m\_p^b(X)$$

$$m\_p^b(Y) \leftarrow m\_p^b(X), t^{bff}(X, Y, Z)$$

$$m\_p^b(Z) \leftarrow m\_p^b(X), t^{bff}(X, Y, Z), \neg p^b(Y)$$

$$p^b(X) \leftarrow m\_p^b(X), t^{bff}(X, Y, Z), \neg p^b(Y), \neg p^b(Z).$$

When a predicate has the unique adornment, we will not display the adornment of the predicate. In the above example, p^b will be denoted by p, and m_p^b by m_p.

DEFINITION 2.13. Some predicates, called IDB (*intensional database*) predicates, are defined by the rules; that is, they appear as the heads of some rules whose bodies are not empty. Other predicates, called EDB (*extensional database*) predicates, are not defined by rules; that is, they always appear as the heads of rules whose bodies are empty. The set of facts that define an EDB predicate is called an EDB set.

[1] In the proofs given in this paper, for simplicity we only consider left-to-right sip strategies in which each lateral has only one incoming arc. Hence, this case will not appear in the paper. However, it can be handled by extending our results.

Since we will assume that rules satisfy range restrictedness, no variables appear in any atom that defines an EDB predicate; that is, any atom in an EDB is ground.

It is practical not to pass bindings to EDB predicates, because EDB predicates are always directly evaluable [6]. But when we prove some properties of magic programs, in order to make the proof simpler, it is helpful not to distinguish between IDB and EDB predicates, and simply to propagate bindings to all predicates.

Let us now turn to the situation in which we need to consider the sip strategy (s2) given in Example 2.2. Kemp *et al.* [14] studied the use of such sip strategies for Datalog$^\neg$. The magic program given in Fig. 2 is obtained by using the sip strategy (s1), but the well-founded model of the magic program does not agree with the well-founded model of the original program in Fig. 1 on the query against the EDB in Fig. 3; that is, $p(a)$ is undefined in the former model, while it is false in the latter. Kemp *et al.* [14] proposed a class of sip strategies, called well-founded sip strategies, that ensures agreement between the well-founded model of the original program and that of the magic program. For instance, if the well-founded sip strategy (s2) is applied to the program in Fig. 1, we have

$$
\begin{aligned}
&m\_p(a)\\
&m\_p(Y) \leftarrow m\_p(X),\ t(X, Y, Z)\\
&m\_p(Z) \leftarrow m\_p(X),\ t(X, Y, Z)\\
&\quad p(X) \leftarrow m\_p(X),\ t(X, Y, Z),\ \neg p(Y),\ \neg p(Z)\\
&\quad p(X) \leftarrow m\_p(X),\ p_0(X).
\end{aligned}
$$

The well-founded model of this magic program agrees on the query with that of the original program. But as we showed in Section 1, all magic facts are true in the well-founded model, and hence it does not improve the performance of the original program.

DEFINITION 2.14. A program is called *safe* if every variable appearing in a negated subgoal also appears in a positive subgoal or in the bound arguments of the head of the rule.

No results presented in this paper require the input program to be safe, but from the practical viewpoint, it is plausible to assume the safety of a given program in order to avoid explicitly materializing the complement of a relation. Actually, we prohibited unsafe usage of negation when we incorporated our method into the Glue–Nail deductive database system [10, 11].

3. MAIN RESULTS

In this section, we formalize the alternating fixpoint technique tailored to magic programs that we introduced intuitively in Section 1.

DEFINITION 3.1. Let MP be the magic program derived from a Datalog$^\neg$ program P in response to a given query by an arbitrary sip strategy. Let Q_α denote the set of all magic facts in O_α. Our idea is to compute the $(\alpha+1)$th underestimate $U_{\alpha+1}$ with Q_α as well as $\overline{O_\alpha}$:

$$
\begin{aligned}
U_0 &= \mathbf{S}_{\mathrm{MP}}(\phi) & O_0 &= \mathbf{S}_{\mathrm{MP}}(\overline{U_0})\\
U_{\alpha+1} &= \mathbf{S}_{\mathrm{MP}\cup Q_\alpha}(\overline{O_\alpha}) & O_{\alpha+1} &= \mathbf{S}_{\mathrm{MP}}(\overline{U_{\alpha+1}}).
\end{aligned}
$$

In what follows, we will loosely refer to U_α as an underestimate and to O_α as an overestimate in order to draw the reader's attention to the similarity of our technique and Van Gelder's technique. Let D_α denote $\{A \in H_P \mid m[A] \in O_\alpha\}$.

Intuitively D_α denotes the set of non-magic facts that are reached via magic facts during the computation of the overestimate O_α. D_α plays an important role in proving the correctness of our method.

EXAMPLE 3.1. Let MP be the magic program shown in Fig. 2 with the EDB presented in Fig. 3. If applied to MP, our alternating fixpoint computes the following underestimates and overestimates:

$$
\begin{aligned}
U_0 &= \{m\_p(a)\}\\
O_0 &= \{p(a), p(b_1), ..., p(b_n), p(c_2), m\_p(a), m\_p(b_1), ...,\\
&\qquad m\_p(b_n), m\_p(c_1), ..., m\_p(c_{n+1})\}\\
U_1 &= \{p(b_n), p(c_2), m\_(a), m\_p(b_1), ...,\\
&\qquad m\_p(b_n), m\_p(c_1), ..., m\_p(c_{n+1})\}\\
O_1 &= \{p(a), p(b_1), p(c_2), m\_p(a), m\_p(b_1), m\_p(c_1),\\
&\qquad m\_(b_2), m\_p(c_2)\}\\
U_2 &= \{p(b_1), p(c_2), m\_p(a), m\_p(b_1), m\_p(c_1),\\
&\qquad m\_p(b_2), m\_p(c_2)\}\\
U_2 &= O_2 = U_3 = O_3 = \cdots.
\end{aligned}
$$

Although each underestimate and overestimate includes all the EDB facts in Fig. 3, for simplicity we omit the EDB. D_0 and D_1 are:

$$
\begin{aligned}
D_0 &= \{p(a), p(b_1), ..., p(b_n), p(c_1), ..., p(c_{n+1})\}\\
D_1 &= \{p(a), p(b_1), p(c_1), p(b_2), p(c_2)\}.
\end{aligned}
$$

We will show that the sequence of overestimates decreases monotonically. Since the original program P is function-free, in a finite number of steps the sequence of overestimates converges to a fixpoint.

DEFINITION 3.2. Suppose that $O_\tau = O_{\tau+1} = \cdots$ for a finite ordinal τ, then $U_{\tau+1} = U_{\tau+2} = \cdots$. We will call $\overline{O_\tau} \cup U_{\tau+1}$ the *magic alternating fixpoint* of MP.

We will show that the magic alternating fixpoint agrees with the well-founded model of the original program P on the query. The most troublesome obstacle to proving this main result is that the sequence of underestimates does not always increase, as shown in Section 1. Furthermore, this non-monotonic nature prevents us from evaluating underestimates in a differential fashion. Fortunately, however, we can prove that any non-magic fact A in U_α appears in $U_{\alpha+1}$ if and only if $m[A]$ is in O_α (that is, $A \in D_\alpha$). Thus, to compute $U_{\alpha+1}$, we can assume all elements in $U_\alpha \cap D_\alpha$ to be proven facts and can ignore any other non-magic facts in U_α. In this way underestimates can be generated in a semi-naive fashion.

Incidentally, recall that when we compute $U_{\alpha+1}$ we use all the magic facts in O_α. We will prove that $U_{\alpha+1} \subseteq O_\alpha$, which implies that while $U_{\alpha+1}$ is being computed no new magic fact will be generated, and therefore we can ignore all rules that define magic predicates.

First, we prove various properties of underestimates and overestimates.

LEMMA 3.1. *For any finite ordinal* α, $U_\alpha \cap D_\alpha \subseteq O_\alpha$ *and* $U_{\alpha+1} \subseteq O_\alpha$.

Proof. We will prove both properties simultaneously by an induction on α. Suppose that $\alpha = 0$.

$$\begin{aligned} U_0 &= \mathbf{S}_{\mathrm{MP}}(\phi) \\ &\subseteq \mathbf{S}_{\mathrm{MP}}(\overline{U_0}) \end{aligned}$$

because $\phi \subseteq \overline{U_0}$ and $\mathbf{S}_{\mathrm{MP}}$ is monotonic.

$$\begin{aligned} &= O_0 \\ U_1 &= \mathbf{S}_{\mathrm{MP} \cup Q_0}(\overline{O_0}) \\ &\subseteq \mathbf{S}_{\mathrm{MP} \cup Q_0}(\overline{U_0}) \end{aligned}$$

because $U_0 \subseteq O_0$, and $\mathbf{S}_{\mathrm{MP}}$ is monotonic.

$$\mathbf{S}_{\mathrm{MP}}(\overline{U_0})$$

since $Q_0 \subseteq \mathbf{S}_{\mathrm{MP}}(\overline{U_0}) = O_0$, apply Proposition 2.2.

$$= O_0.$$

Next, suppose that $\alpha > 0$. We will prove $U_\alpha \cap D_\alpha \subseteq O_\alpha$ by showing that

$$\mathbf{T}_{\mathrm{MP} \cup Q_{\alpha-1}}[\overline{O_{\alpha-1}}] \uparrow \beta \cap D_\alpha \subseteq O_\alpha,$$

by an induction on β. If $\beta = 0$ the proof is trivial, so let us assume that $\beta > 0$. Select an arbitrary element, say A, from the left-hand side. Then, there exists the following ground instance of a rule in MP:

$$A \leftarrow m[A], L_1, ..., L_k.$$

In order to prove that $A \in O_\alpha$, we will show that each subgoal in the body appears in $O_\alpha \cup \overline{U_\alpha}$, because $O_\alpha = \mathbf{S}_{\mathrm{MP}}(\overline{U_\alpha})$. First, $m[A] \in O_\alpha$, because $A \in D_\alpha$. Second, if L_i is negative, $L_i \in \overline{O_{\alpha-1}}$. From the inductive hypothesis of α, $U_\alpha \subseteq O_{\alpha-1}$, and hence $L_i \in \overline{U_\alpha}$. Next, if L_i is positive, $L_i \in \mathbf{T}_{\mathrm{MP} \cup Q_{\alpha-1}}[\overline{O_{\alpha-1}}] \uparrow (\beta - 1)$. From the inductive hypothesis of β,

$$\mathbf{T}_{\mathrm{MP} \cup Q_{\alpha-1}}[\overline{O_{\alpha-1}}] \uparrow (\beta - 1) \cap D_\alpha \subseteq O_\alpha. \tag{1}$$

We claim that $L_i \in D_\alpha$ for the purpose of proving that $L_i \in O_\alpha$. Observe that for each $i = 1, ..., k$, there exists the following ground instance of a rule in MP:[2]

$$m[L_i] \leftarrow m[A], L_1, ..., L_{i-1}.$$

Since $m[A] \in O_\alpha$, using $m[L_1] \leftarrow m[A]$ we can see that $m[L_1] \in O_\alpha$. Thus, if L_1 is positive, $L_1 \in D_\alpha$, and hence $L_1 \in O_\alpha$ from (1). Next, since L_1 is in O_α or in $\overline{U_\alpha}$, using $m[L_2] \leftarrow m[A], L_1$, we prove that $m[L_2] \in O_\alpha$. If L_2 is positive, $L_2 \in D_\alpha$. We repeat this process to prove that $L_i \in D_\alpha$ for each $i = 1, ..., k$. This completes the proof of the property that $U_\alpha \cap D_\alpha \subseteq O_\alpha$.

Next, we prove that $U_{\alpha+1} \subseteq O_\alpha$ by showing that

$$\mathbf{T}_{\mathrm{MP} \cup Q_\alpha}[\overline{O_\alpha}] \uparrow \beta \subseteq O_\alpha,$$

by an induction on β. When $\beta = 0$, the result is trivial, so let us assume that $\beta > 0$. Let A be an arbitrary element in the left-hand side. We assume that A is a non-magic atom, but the case when A is a magic fact can be proved in a similar way. There exists the following ground instance of a rule in MP:

$$A \leftarrow m[A], L_1, ..., L_k.$$

For the purpose of proving that $A \in O_\alpha$, we claim that each subgoal in the body is in O_α or in $\overline{U_\alpha}$. First, $m[A] \in O_\alpha$ and $L_i \in O_\alpha$ if L_i is positive, because $\mathbf{T}_{\mathrm{MP} \cup Q_\alpha}[\overline{O_\alpha}] \uparrow (\beta - 1) \subseteq O_\alpha$ from the inductive hypothesis of β. Next, if L_i is negative, $L_i \in \overline{O_\alpha}$. Since we have proved that $U_\alpha \cap D_\alpha \subseteq O_\alpha$, from Proposition 2.1 we have

$$\overline{O_\alpha} \cap \neg \cdot D_\alpha \subseteq \overline{U_\alpha}. \tag{2}$$

[2] Throughout the following proofs we will assume for brevity that MP is derived by left-to-right sip strategies. All results can be easily extended to any sip strategies. Furthermore, for simplicity, we do not distinguish between EDB and IDB predicates, and we provide magic predicates for EDB predicates as well as for IDB predicates.

To prove that $L_i \in \overline{U_\alpha}$, it remains for us to show that $L_i \in \neg \cdot D_\alpha$. Note that, for each $i = 1, ..., k$, there exist the following ground instance of a rule in MP:

$$m[L_i] \leftarrow m[A], L_1, ..., L_{i-1}.$$

Since $m[A] \in O_\alpha$, from the rule $m[L_1] \leftarrow m[A]$ we have $m[L_1] \in O_\alpha$. Thus, if L_1 is negative, $L_1 \in \neg \cdot D_\alpha$, and hence $L_1 \in \overline{U_\alpha}$ by (2). Next, since L_1 is in O_α or in $\overline{U_\alpha}$, using the rule $m[L_2] \leftarrow m[A], L_1$ we see that $m[L_2] \in O_\alpha$. Again, if L_2 is negative, $L_2 \in \neg \cdot D_\alpha$. We repeat this process and prove that, if L_i is negative, $L_i \in \neg \cdot D_\alpha$. ∎

LEMMA 3.2. *For any finite ordinal* α, $U_\alpha \cap D_\alpha \subseteq \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})$.

Proof. We will prove that

$$\mathbf{T}_{\mathrm{MP} \cup Q_{\alpha-1}}[\overline{O_{\alpha-1}}] \uparrow \beta \cap D_\alpha \subseteq \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}}),$$

by an induction on β. The proof is trivial when $\beta = 0$. Consider the case when $\beta > 0$ and let A be an arbitrary element in the left-hand side. There exists the following ground instance of a rule in MP:

$$A \leftarrow m[A], L_1, ..., L_k.$$

In order to prove $A \in \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})$, we will show that each subgoal in the body appears in $\mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})$ or in $\overline{O_{\alpha-1}}$. First, $m[A] \in \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})$. This is because $A \in D_\alpha$, $m[A] \in O_\alpha$, and hence $m[A] \in Q_\alpha$. Note that $Q_\alpha \subseteq \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})$. Next, if L_i is negative, $L_i \in \overline{O_{\alpha-1}}$. Furthermore, since $U_\alpha \subseteq O_{\alpha-1}$ from Lemma 3.1, L_i is in $\overline{U_\alpha}$, which we will use later in this proof. Finally, if L_i is positive, $L_i \in U_\alpha$ and $L_i \in \mathbf{T}_{\mathrm{MP} \cup Q_{\alpha-1}}[\overline{O_{\alpha-1}}] \uparrow (\beta - 1)$. From the inductive hypothesis of β,

$$\mathbf{T}_{\mathrm{MP} \cup Q_{\alpha-1}}[\overline{O_{\alpha-1}}] \uparrow (\beta - 1) \cap D_\alpha \subseteq \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}}).$$

Thus we will show that $L_i \in D_\alpha$ in order to prove that $L_i \in \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})$. For each $i = 1, ..., k$, there exists the following ground instance of a rule in MP:

$$m[L_i] \leftarrow m[A], L_1, ..., L_{i-1}.$$

Since $m[A] \in O_\alpha$, from the rule $m[L_1] \leftarrow m[A]$ we can see that $m[L_1] \in O_\alpha$, and hence $L_1 \in D_\alpha$ if L_1 is positive. Furthermore, recall that $L_1 \in U_\alpha$. Since $U_\alpha \cap D_\alpha \subseteq O_\alpha$ from Lemma 3.1, $L_1 \in O_\alpha$. Next, since L_1 is in O_α or $\overline{U_\alpha}$, from the rule $m[L_2] \leftarrow m[A], L_1$ we have $m[L_2] \in O_\alpha$. We repeat this reasoning and prove that $L_i \in O_\alpha \cup \overline{U_\alpha}$ and $m[L_i] \in O_\alpha$ for each $i = 1, ..., k$, and therefore that $L_i \in D_\alpha$ if L_i is positive.

LEMMA 3.3. *For any finite ordinal* α, $O_{\alpha+1} \subseteq O_\alpha$.

Proof. We will prove the property by an induction on α. When $\alpha = 0$, we have

$$\begin{aligned} U_0 &= \mathbf{S}_{\mathrm{MP}}(\phi) \\ &\subseteq \mathbf{S}_{\mathrm{MP}}(\overline{O_0}) \end{aligned}$$

because $\phi \subseteq \overline{O_0}$, and $\mathbf{S}_{\mathrm{MP}}$ is monotonic

$$= \mathbf{S}_{\mathrm{MP} \cup Q_0}(\overline{O_0})$$

since $Q_0 \subseteq \mathbf{S}_{\mathrm{MP}}(\overline{O_0})$, apply Proposition 2.2

$$\begin{aligned} &= U_1 \\ O_1 &= \mathbf{S}_{\mathrm{MP}}(\overline{U_1}) \\ &\subseteq \mathbf{S}_{\mathrm{MP}}(\overline{U_0}) \end{aligned}$$

because $U_0 \subseteq U_1$ from Lemma 3.1, and $\mathbf{S}_{\mathrm{MP}}$ is monotonic

$$= O_0.$$

Next, suppose that $\alpha > 0$. Then we have:

$$O_{\alpha+1} = \mathbf{S}_{\mathrm{MP}}(\overline{\mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_\alpha})})$$

from the definition of $O_{\alpha+1}$.

$$\subseteq \mathbf{S}_{\mathrm{MP}}(\overline{\mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})})$$

$O_\alpha \subseteq O_{\alpha-1}$ from the inductive hypothesis of α, and $\mathbf{S}_{\mathrm{MP}}$ is monotonic. It remains for us to prove that

$$\mathbf{S}_{\mathrm{MP}}(\overline{\mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})}) \subseteq O_\alpha.$$

We will show that

$$\mathbf{T}_{\mathrm{MP}}[\overline{\mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})}] \uparrow \beta \subseteq O_\alpha,$$

by an induction on β. When $\beta = 0$ the proof is trivial. Consider the case in which $\beta > 0$, and let A be an arbitrary element in the left-hand side. We assume that A is a non-magic atom, but the arguments carry over to the case in which A is a magic fact. There exists the following ground instance of a rule in MP:

$$A \leftarrow m\_A, L_1, ..., L_k.$$

In order to prove that $A \in O_\alpha$, we will prove that each subgoal in the body appears in O_α or in $\overline{U_\alpha}$. First, $m[A] \in O_\alpha$ and $L_i \in O_\alpha$ if L_i is positive, because from the inductive hypothesis of β,

$$\mathbf{T}_{\mathrm{MP}}[\overline{\mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})}] \uparrow (\beta - 1) \subseteq O_\alpha.$$

If L_i is negative, $L_i \in \overline{\mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})}$. From Lemma 3.2, $U_\alpha \cap D_\alpha \subseteq \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})$. From Proposition 2.1,

$$\overline{\mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})} \cap \neg \cdot D_\alpha \subseteq \overline{U_\alpha}. \tag{3}$$

Thus, in order to prove that $L_i \in \overline{U_\alpha}$, we will show $L_i \in \neg \cdot D_\alpha$. Note that, for each $i = 1, ..., k$, there exists the following ground instance of a rule in MP:

$$m[L_i] \leftarrow m[A], L_1, ..., L_{i-1}.$$

Since $m[A] \in O_\alpha$, from the rule $m[L_1] \leftarrow m[A]$ we can see that $m[L_1] \in O_\alpha$. If L_1 is negative, $L_1 \in \neg \cdot D_\alpha$. From (3), we have $L_1 \in \overline{U_\alpha}$. Next, consider the rule $m[L_2] \leftarrow m[A], L_1$. Since $L_1 \in O_\alpha \cup \overline{U_\alpha}$, we have $m[L_2] \in O_\alpha$. By repeating this reasoning, we can prove that $L_i \in O_\alpha \cup \overline{U_\alpha}$ and $m[L_i] \in O_\alpha$ for each $i = 1, ..., k$, and hence $L_i \in \neg \cdot D_\alpha$ if L_i is negative. ∎

LEMMA 3.4. *For any finite ordinal* α, $U_\alpha \cap D_\alpha \subseteq U_{\alpha+1}$.

Proof.

$$U_\alpha \cap D_\alpha \subseteq \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_{\alpha-1}})$$

from Lemma 3.2

$$\subseteq \mathbf{S}_{\mathrm{MP} \cup Q_\alpha}(\overline{O_\alpha})$$

because $O_{\alpha-1} \supseteq O_\alpha$ from Lemma 3.3, and $\mathbf{S}_{\mathrm{MP}}$ is monotonic

$$= U_{\alpha+1}. \quad ∎$$

LEMMA 3.5. *For any finite ordinal* α, $((U_\alpha \cap H_P) - D_\alpha) \cap U_{\alpha+1} = \phi$.

Proof. We will prove the following inequality, because it implies the above equation

$$U_{\alpha+1} \cap H_P \subseteq D_\alpha.$$

Select an arbitrary element, say A, from the left-hand side. There must exist a ground instance of a rule of MP such that the head is A and the body includes $m[A]$. $m[A]$ is in $U_{\alpha+1}$. Since $U_{\alpha+1} \subseteq O_\alpha$ from Lemma 3.1, $m[A]$ is in O_α, and therefore A appears in D_α. ∎

We will now show that the magic alternating fixpoint gives the correct answers to the query with respect to the well-founded model of the original program.

LEMMA 3.6. *Let P be a Datalog*$^\neg$ *program. Suppose that* $\mathbf{W}_P \uparrow \sigma = \mathbf{W}_P \uparrow (\sigma+1)$ *for a finite ordinal* σ, *and that* $O_\tau = O_{\tau+1}$ *for a finite ordinal* τ. *Then we have*

$$(\mathbf{W}_P \uparrow \sigma)^+ \cap D_\tau \subseteq U_{\tau+1}$$
$$(\mathbf{W}_P \uparrow \sigma)^- \subseteq \overline{O_\tau}.$$

Proof. We will simultaneously show the following two propositions by an induction on α ($\leqslant \sigma$):

$$(\mathbf{W}_P \uparrow \alpha)^+ \cap D_\tau \subseteq U_{\tau+1} \tag{4}$$
$$(\mathbf{W}_P \uparrow \alpha)^- \subseteq \overline{O_\tau}. \tag{5}$$

The proof is trivial when $\alpha = 0$, so suppose that $\alpha > 0$. We will prove (4) first. Let A be an arbitrary element in $(\mathbf{W}_P \uparrow \alpha)^+ \cap D_\tau$. Since $(\mathbf{W}_P \uparrow \alpha)^+ = \mathbf{T}_P(\mathbf{W}_P \uparrow (\alpha-1))$, there exists the following ground instance of a rule in P:

$$A \leftarrow L_1, ..., L_k,$$

where each subgoal appears in $\mathbf{W}_P \uparrow (\alpha-1)$. Thus there also exists

$$A \leftarrow m[A], L_1, ..., L_k. \tag{6}$$

In order to prove that $A \in U_{\tau+1}$, since $U_{\tau+1} = \mathbf{S}_{\mathrm{MP} \cup Q_\tau}(\overline{O_\tau})$, we will show that each subgoal in the body of (6) appears in $U_{\tau+1}$ or in $\overline{O_\tau}$. First, $m[A] \in U_{\tau+1}$, because from $A \in D_\tau$ we have $m[A] \in O_\tau$, and hence $m[A] \in Q_\tau$. Next, if L_i is negative, $L_i \in (\mathbf{W}_P \uparrow (\alpha-1))^-$. From the inductive hypothesis of α, $(\mathbf{W}_P \uparrow (\alpha-1))^- \subseteq \overline{O_\tau}$, and hence

$$L_i \in \overline{O_\tau}. \tag{7}$$

Next, if L_i is positive, $L_i \in (\mathbf{W}_P \uparrow (\alpha-1))^+$. From the inductive hypothesis of α,

$$(\mathbf{W}_P \uparrow (\alpha-1))^+ \cap D_\tau \subseteq U_{\tau+1}. \tag{8}$$

In order to prove that $L_i \in U_{\tau+1}$, it remains for us to prove that $L_i \in D_\tau$. Note that for each $i = 1, ..., k$, we have

$$m[L_i] \leftarrow m[A], L_1, ..., L_{i-1}. \tag{9}$$

We will prove that $m[L_i] \in O_\tau$, which immediately implies that $L_i \in D_\tau$ if L_i is positive. Since $O_\tau = \mathbf{S}_{\mathrm{MP}}(\overline{U_\alpha})$, we show that each subgoal in the body of (9) appears in $O_\tau \cup \overline{U_\alpha}$. Since $m[A] \in O_\tau$, we have $m[L_1] \in O_\tau$ from the rule $m[L_1] \leftarrow m[A]$.

- If L_1 is positive, $L_1 \in D_\tau$. From (8), $L_1 \in U_{\tau+1}$. From Lemma 3.1, $U_{\tau+1} \subseteq O_\tau$, and hence $L_1 \in O_\tau$.
- If L_1 is negative, $L_1 \in \neg \cdot D_\tau$. From (7), $L_1 \in \overline{O_\tau}$. From Lemma 3.1 $U_\tau \cap D_\tau \subseteq O_\tau$. From Proposition 2.1, $\overline{O_\tau} \cap \neg \cdot D_\tau \subseteq \overline{U_\tau}$. Thus $L_1 \in \overline{U_\tau}$.

Since $m[A] \in O_\tau$ and L_1 is in O_τ or in $\overline{U_\tau}$ as shown above, from the rule $m[L_2] \leftarrow m[A], L_1$ we can see that $m[L_2] \in O_\tau$. We repeat this process and can prove $m[L_i] \in O_\tau$ for each $i = 1, ..., k$. This completes the proof of (4).

Next, in order to prove (5), we will first show that

$$\mathbf{S}_P(\overline{(\mathbf{W}_P \uparrow (\alpha-1))^+}) \supseteq \mathbf{S}_{\mathrm{MP}}(\overline{U_{\tau+1}}) \cap H_P. \tag{10}$$

Observe that

$$\mathbf{S}_P(\overline{(\mathbf{W}_P \uparrow (\alpha-1))^+}) \supseteq \mathbf{S}_{\mathrm{MP}}(\overline{(\mathbf{W}_P \uparrow (\alpha-1))^+}) \cap H_P, \tag{11}$$

because if supplied with the same set of negative literals, any non-magic fact inferred in MP is also derived in P. From the inductive hypothesis of α, $(\mathbf{W}_P \uparrow (\alpha-1))^+ \cap D_\tau \subseteq U_{\tau+1}$. From Proposition 2.1 and $D_\tau = d_{\tau+1}$, we have

$$\overline{(\mathbf{W}_P \uparrow (\alpha-1))^+} \supseteq \overline{U_{\tau+1}} \cap \neg \cdot D_{\tau+1}. \tag{12}$$

Since $\mathbf{S}_{\mathrm{MP}}$ is monotonic, it follows from (12) that

$$\mathbf{S}_{\mathrm{MP}}(\overline{(\mathbf{W}_P \uparrow (\alpha-1))^+}) \supseteq \mathbf{S}_{\mathrm{MP}}(\overline{U_{\tau+1}} \cap \neg \cdot D_{\tau+1}). \tag{13}$$

Furthermore, we have

$$\mathbf{S}_{\mathrm{MP}}(\overline{U_{\tau+1}} \cap \neg \cdot D_{\tau+1}) = \mathbf{S}_{\mathrm{MP}}(\overline{U_{\tau+1}}), \tag{14}$$

because, during the computation of $\mathbf{S}_{\mathrm{MP}}(\overline{U_{\tau+1}})$, if any negative fact L in $\overline{U_{\tau+1}}$ is used to do some inference, $m[L]$ must have been inferred and is therefore in $\mathbf{S}_{\mathrm{MP}}(\overline{U_{\tau+1}})(= O_{\tau+1})$. Thus L is also in $\neg \cdot D_{\tau+1}$. We can show that (10) holds from (11), (13), and (14). We will now derive (5) from (10). From Theorem 2.4,

$$\mathbf{S}_P(\overline{(\mathbf{W}_P \uparrow (\alpha-1))^+}) = H_P - \mathbf{U}_P(\mathbf{W}_P \uparrow (\alpha-1)).$$

Since $O_\tau = O_{\tau+1} = \mathbf{S}_{\mathrm{MP}}(\overline{U_{\tau+1}})$, from (10) we have

$$H_P - \mathbf{U}_P(\mathbf{W}_P \uparrow (\alpha-1)) \supseteq O_\tau \cap H_P.$$

This inequality implies that

$$\mathbf{U}_P(\mathbf{W}_P \uparrow (\alpha-1)) \subseteq H_P - O_\tau.$$

Since $\neg \cdot \mathbf{U}_P(\mathbf{W}_P \uparrow (\alpha-1)) = (\mathbf{W}_P \uparrow \alpha)^-$ and $H_P \subseteq H_{\mathrm{MP}}$,

$$(\mathbf{W}_P \uparrow \alpha)^- \subseteq \neg \cdot (H_{\mathrm{MP}} - O_\tau) = \overline{O_\tau},$$

which completes the proof of (5). ∎

LEMMA 3.7. *Let P be a Datalog$^\neg$ program. Suppose that $\mathbf{W}_P \uparrow \sigma = \mathbf{W}_P \uparrow (\sigma+1)$ for a finite ordinal σ, and that $O_\tau = O_{\tau+1}$ for a finite ordinal τ. Then we have*:

$$(\mathbf{W}_P \uparrow \sigma)^+ \supseteq U_\tau \cap D_\tau$$
$$(\mathbf{W}_P \uparrow \sigma)^- \supseteq \overline{O_\tau} \cap \neg \cdot D_\tau.$$

Proof. We will show the following two propositions simultaneously by an induction on β ($\leqslant \tau$)

$$(\mathbf{W}_P \uparrow \sigma)^+ \supseteq U_\beta \cap D_\beta \tag{15}$$
$$(\mathbf{W}_P \uparrow \sigma)^- \supseteq \overline{O_\beta} \cap \neg \cdot D_\beta \tag{16}$$

When $\beta = 0$, (15) is trivially true. Suppose that $\beta > 0$. Later we will show that (15) implies (16) for any β. We will now show (15) by assuming that

$$(\mathbf{W}_P \uparrow \sigma)^- \supseteq \overline{O_{\beta-1}} \cap \neg \cdot D_{\beta-1}. \tag{17}$$

Equation (15) can be shown by proving that, for any finite ordinal γ,

$$(\mathbf{W}_P \uparrow \sigma)^+ \supseteq \mathbf{T}_{\mathrm{MP} \cup Q_{\beta-1}}[\overline{O_{\beta-1}}] \uparrow \gamma \cap D_\beta,$$

by an induction on γ. The case in which $\gamma = 0$ is trivial, so let us assume that $\gamma > 0$. Let A be an arbitrary element in the right-hand side. There exists the following ground instance of a rule in MP:

$$A \leftarrow m[A], L_1, ..., L_k, \tag{18}$$

where each positive subgoal appears in $\mathbf{T}_{\mathrm{MP} \cup Q_{\beta-1}}[\overline{O_{\beta-1}}] \uparrow (\gamma-1)$ and each negative subgoal is in $\overline{O_{\beta-1}}$. Thus, there also exists the following ground instance of a rule in P:

$$A \leftarrow L_1, ..., L_k.$$

To prove that $A \in (\mathbf{W}_P \uparrow \sigma)^+$, since $\mathbf{W}_P \uparrow \sigma$ is a fixpoint of $\mathbf{W}_P$, we show that each L_i in the body also appears in $\mathbf{W}_P \uparrow \sigma$. Observe that, for each $i = 1, ..., k$, there also exists

$$m[L_i] \leftarrow m[A], L_1, ..., L_{i-1}.$$

Since $A \in D_\beta$, it follows that $m[A] \in O_\beta$. From the rule $m[L_1] \leftarrow m[A]$, we can see that $m[L_1] \in O_\beta$.

- If L_1 is positive, $L_1 \in D_\beta$. Since L_1 appears in the body of (18), $L_1 \in \mathbf{T}_{\mathrm{MP} \cup Q_{\beta-1}}[\overline{O_{\beta-1}}] \uparrow (\gamma-1) \subseteq U_\beta$. From the inductive hypothesis of γ, $L_1 \in (\mathbf{W}_P \uparrow \sigma)^+$. Incidentally, since $L_1 \in U_\beta$ and $U_\beta \cap D_\beta \subseteq O_\beta$ from Lemma 3.1, we have that $L_1 \in O_\beta$.
- If L_1 is negative, $L_1 \in \neg \cdot D_\beta$. L_1 also appears in $\neg \cdot D_{\beta-1}$, because $O_\beta \subseteq O_{\beta-1}$ from Lemma 3.3. Since L_1 appears in the body of (18), $L_1 \in \overline{O_{\beta-1}}$. From (17), L_1 is in $(\mathbf{W}_P \uparrow \sigma)^-$. Furthermore, since $L_1 \in \overline{O_{\beta-1}}$ and $U_\beta \subseteq O_{\beta-1}$ from Lemma 3.1, we have that $L_1 \in \overline{U_\beta}$.

We have proved that $m[A] \in O_\beta$ and that L_1 appears in O_β or in $\overline{U_\beta}$. From $m[L_2] \leftarrow m[A], L_1$ we have $m[L_2] \in O_\beta$. By repeating this reasoning, we can prove that each L_i is in $\mathbf{W}_P \uparrow \sigma$. This completes the proof of (15).

We will now prove (16) by assuming (15). First observe the following equations:

$$(\mathbf{W}_P \uparrow \sigma)^- = (\mathbf{W}_P \uparrow (\sigma+1))^-$$

because $\mathbf{W}_P \uparrow \sigma$ is a fixpoint of $\mathbf{W}_P$

$$= \neg \cdot \mathbf{U}_P(\mathbf{W}_P \uparrow \sigma)$$
$$= \neg \cdot (H_P - (H_P - \mathbf{U}_P(\mathbf{W}_P \uparrow \sigma)))$$
$$= \overline{H_P - \mathbf{U}_P(\mathbf{W}_P \uparrow \sigma)}.$$

Thus, (16) is equivalent to

$$\overline{H_P - \mathbf{U}_P(\mathbf{W}_P \uparrow \sigma)} \supseteq \overline{O_\beta} \cap \neg \cdot D_\beta.$$

From Proposition 2.1, the above property is implied by

$$(H_P - \mathbf{U}_P(\mathbf{W}_P \uparrow \sigma)) \cap D_\beta \subseteq O_\beta.$$

From Theorem 2.4, the above property is equivalent to

$$\mathbf{S}_P(\overline{(\mathbf{W}_P \uparrow \sigma)^+}) \cap D_\beta \subseteq O_\beta.$$

We will therefore prove that, for any finite ordinal γ,

$$\mathbf{T}_P[\overline{\mathbf{W}_P \uparrow \sigma)^+}] \uparrow \gamma \cap D_\beta \subseteq O_\beta$$

by an induction on γ. Let A be an arbitrary element in the left-side. Then, there exists a ground instance of a rule in P,

$$A \leftarrow L_1, ..., L_k, \tag{19}$$

where each positive subgoal in the body is in $\mathbf{T}_P[\overline{(\mathbf{W}_P \uparrow \sigma)^+}] \uparrow (\gamma - 1)$ and each negative subgoal is in $\overline{(\mathbf{W}_P \uparrow \sigma)^+}$. Thus, there exists the following ground instance of a rule in MP

$$A \leftarrow m[A], L_1, ..., L_k.$$

To show that $A \in O_\beta$, we will show that each positive subgoal in the body of the above rule is in O_β and each negative one is in $\overline{U_\beta}$. First, $m[A] \in O_\beta$, because $A \in D_\beta$. Next, observe that for each $i = 1, ..., k$, there exists

$$m[L_i] \leftarrow m[A], L_1, ..., L_{i-1}.$$

Since $m[A] \in O_\beta$, we have $m[L_1] \in O_\beta$ from the rule $m[L_1] \leftarrow m[A]$.

- If L_1 is positive, $L_1 \in D_\beta$. Since L_1 is in the body of (19), from the inductive hypothesis of γ, $L_1 \in O_\beta$.
- If L_1 is negative, $L_1 \in \neg \cdot D_\beta$. Further, since L_1 is in the body of (19), $L_1 \in \overline{(\mathbf{W}_P \uparrow \sigma)^+}$. From the application of Proposition 2.1 to (15), it follows that $\overline{(\mathbf{W}_P \uparrow \sigma)^+} \cap \neg \cdot D_\beta \subseteq \overline{U_\beta}$. Thus we have $L_1 \in \overline{U_\beta}$.

Since $m[A] \in O_\beta$ and $L_1 \in O_\beta \cup \overline{U_\beta}$, from the rule $m[L_2] \leftarrow m[A], L_1$ we can see that $m[L_2] \in O_\beta$. By repeating this process, we can prove that each L_i is in O_β or in $\overline{U_\beta}$. ∎

From Lemmas 3.6 and 3.7 we have the following theorem, which ensures the correctness of our alternating fixpoint technique for magic programs.

THEOREM 3.1. *Let P be a Datalog$^\neg$ program. Suppose that $\mathbf{W}_P \uparrow \sigma = \mathbf{W}_P \uparrow (\sigma+1)$ for a finite ordinal α, and that $O_\tau = O_{\tau+1}$ for a finite ordinal τ. Then, we have*

$$(\mathbf{W}_P \uparrow \sigma)^+ \cap D_\tau = U_{\tau+1} \cap D_\tau$$
$$(\mathbf{W}_P \uparrow \sigma)^- \cap \neg \cdot D_\tau = \overline{O_\tau} \cap \neg \cdot D_\tau.$$

Since D_τ includes the query, the well-founded model of P and $\overline{O_\tau} \cup U_{\tau+1}$ agree on the query.

4. RELATED WORKS

4.1. *Kemp et al.'s Technique*

First, we will compare our method with Kemp *et al.*'s method [15]. Kemp *et al.* showed the following result for general programs that may not be modularly stratified.

THEOREM 4.1 [15]. *Let P be a Datalog$^\neg$ program, and let* MP *be the magic program derived from P by an arbitrary sip strategy. Let $\mathscr{M}$ denote the set of all magic facts that are true or undefined in the well-founded model of* MP. *Then, the well-founded model of P agrees with that of* MP $\cup \mathscr{M}$ *on the query.*

The above method uses the fixed set of magic facts, namely $\mathscr{M}$, during the computation. We will show that the set of all magic facts in the final overestimate generated by our magic alternating fixpoint technique is always a subset of $\mathscr{M}$. Recall that, in the case of the motivating example of Section 1.1, the former set is significantly smaller than the latter.

THEOREM 4.2. *Let* MP *be a magic program. Let $\mathscr{M}$ be the set of all magic facts that are true or undefined in the well-founded model of* MP. *The set of all magic facts in the final overestimate of the magic alternating fixpoint computation is a subset of $\mathscr{M}$.*

Proof. First, let us compute $\mathscr{M}$ by Van Gelder's alternating fixpoint. Let $\mathscr{U}_\alpha$ and $\mathscr{O}_\alpha$ denote respectively an

underestimate and an overestimate of Van Gelder's alternating fixpoint applied to MP:

$$\mathscr{U}_0 = \mathbf{S}_{\mathrm{MP}}(\phi) \qquad \mathscr{O}_0 = \mathbf{S}_{\mathrm{MP}}(\overline{\mathscr{U}_0})$$
$$\mathscr{U}_{\alpha+1} = \mathbf{S}_{\mathrm{MP}}(\overline{\mathscr{O}_\alpha}) \qquad \mathscr{O}_{\alpha+1} = \mathbf{S}_{\mathrm{MP}}(\overline{\mathscr{U}_{\alpha+1}}).$$

Suppose that $\mathscr{O}_\sigma$ is the final overestimate; in other words, that $\mathscr{O}_\sigma = \mathscr{O}_{\sigma+1}$ for a finite ordinal σ. From Theorem 2.5, $\mathscr{O}_\alpha$ is the set of facts that are true or undefined in the well-founded model of MP, and hence the set of all magic facts in $\mathscr{O}_\sigma$ is equal to $\mathscr{M}$.

Let U_α and O_α denote an underestimate and an overestimate of the magic alternating fixpoint applied to MP, which are defined in Definition 3.1:

$$U_0 = \mathbf{S}_{\mathrm{MP}}(\phi) \qquad O_0 = \mathbf{S}_{\mathrm{MP}}(\overline{U_0})$$
$$U_{\alpha+1} = \mathbf{S}_{\mathrm{MP}\cup Q_\alpha}(\overline{O_\alpha}) \qquad O_{\alpha+1} = \mathbf{S}_{\mathrm{MP}}(\overline{U_{\alpha+1}}).$$

We will show that, for any finite ordinal α, $U_\alpha \supseteq \mathscr{U}_\alpha$ and $O_\alpha \subseteq \mathscr{O}_\alpha$. The proof is an induction on α. When $\alpha = 0$, $U_0 = \mathbf{S}_{\mathrm{MP}}(\phi) = \mathscr{U}_0$, which implies $O_0 = \mathscr{O}_0$. Suppose that $\alpha > 0$. From the inductive hypothesis, $O_{\alpha-1} \subseteq \mathscr{O}_{\alpha-1}$. Thus, we have $U_\alpha \supseteq \mathscr{U}_\alpha$, because

$$U_\alpha = \mathbf{S}_{\mathrm{MP}\cup Q_{\alpha-1}}(\overline{O_{\alpha-1}}) \supseteq \mathbf{S}_{\mathrm{MP}}(\overline{O_{\alpha-1}}) \supseteq \mathbf{S}_{\mathrm{MP}}(\overline{\mathscr{O}_{\alpha-1}}) = \mathscr{U}_\alpha.$$

It immediately follows that $O_\alpha \subseteq \mathscr{O}_\alpha$.

Let O_α be the final overestimate of the magic alternating fixpoint computation, that is, let $O_\tau = O_{\tau+1}$. Let η be the maximum number of σ and τ. Then, $O_\tau = O_\eta$, $\mathscr{O}_\sigma = \mathscr{O}_\eta$, and hence $O_\tau = O_\eta \subseteq \mathscr{O}_\eta = \mathscr{O}_\sigma$. It follows that the set of all magic facts in O_τ is a subset of $\mathscr{M}$. ∎

From Theorem 4.2, in the final step the magic alternating fixpoint generates fewer magic facts than Kemp *et al.*'s technique assumes. This, however, does not always imply that the former technique is more efficient than Kemp *et al.*'s. The following example is helpful for understanding this situation.

EXAMPLE 4.1. Consider the magic program MP in Fig. 2 with the following EDB:

$$p_0(c_1) \quad t(a, c_0, b_1) \quad t(b_1, c_1, b_2)$$
$$t(b_2, c_2, b_3) \quad \cdots \quad t(b_n, c_n, c_{n+1}).$$

Kemp *et al.*'s method in Theorem 4.1 first computes $\mathscr{M}$, which is the set of all magic facts in the well-founded model of MP. Suppose that we have already obtained $\mathscr{M}$, which is $\{m\_p(a), m\_p(c_0), m\_p(b_1), m\_p(c_1)\}$. Then, Kemp *et al.*'s technique proceeds to calculate the well-founded model of MP $\cup$ $\mathscr{M}$. Let us compute it by using Van Gelder's alternating fixpoint:

$$\mathscr{U}_0 = \{m\_p(a), m\_p(b_1), m\_p(c_0), m\_p(c_1), p(c_1)\}$$
$$\mathscr{O}_0 = \{m\_p(a), m\_p(b_1), m\_p(c_0), m\_p(c_1), p(a), p(c_1)\}$$
$$\mathscr{U}_1 = \mathscr{O}_1.$$

Next let us compute the magic alternating fixpoint:

$$U_0 = \{m\_p(a), m\_p(c_0)\}$$
$$O_0 = \{m\_p(a), m\_p(b_1), ..., m\_p(b_{n+1}), m\_p(c_0), ..., m\_p(c_n), p(a), p(b_1), ..., p(b_n), p(c_1)\}$$
$$U_1 = \{m\_p(a), m\_p(b_1), ..., m\_p(b_{n+1}), m\_p(c_0), ..., m\_p(c_n), p(c_1)\}$$
$$O_1 = \{m\_p(a), m\_p(b_1), m\_p(c_0), m\_p(c_1), p(a), p(c_1)\}$$
$$U_2 = O_1.$$

The former computation takes fewer steps to reach a fixpoint than the latter. Thus, we may conclude that in this case Kemp *et al.*'s technique is superior to the magic alternating fixpoint if we ignore the cost of computing $\mathscr{M}$. Now let us look at the cost of generating $\mathscr{M}$. As indicated in the proof of Theorem 4.2, $\mathscr{M}$ is the set of all magic facts that appear in the final overestimate of Van Gelder's alternating fixpoint applied to MP. The following sequence shows the application of Van Gelder's technique to MP:

$$\mathscr{U}'_0 = \{m\_p(a), m\_p(c_0)\}$$
$$\mathscr{O}'_0 = \{m\_p(a), m\_p(b_1), ..., m\_p(b_{n+1}), m\_p(c_0), ..., m\_p(c_n), p(a), p(b_1), ..., p(b_n), p(c_1)\}$$
$$\mathscr{U}'_1 = \{m\_p(a), m\_p(b_1), m\_p(c_0), m\_p(c_1), p(c_1)\}$$
$$\mathscr{O}'_1 = \{m\_p(a), m\_p(b_1), m\_p(c_0), m\_p(c_1), p(a), p(c_1)\}$$
$$\mathscr{U}'_2 = \mathscr{O}'_1.$$

In view of the above computation, the cost of Kemp *et al.*'s method is almost equal to that of the magic alternating fixpoint technique.

4.2. *Techniques for Modularly Stratified Programs*

Since any modularly stratified program has a two-valued well-founded model, we only need to compute its positive facts. This property has been used as a basis for several proposed methods [14, 15, 20, 23] of handling modularly stratified programs. On the other hand, the magic alternating fixpoint technique was created for non-modularly stratified programs that may have three-valued well-founded models, and hence, in addition to true facts (underestimates), it also has to compute undefined facts (overestimates). Owing to this extra burden, if applied to modularly stratified programs, the magic alternating fixpoint technique could be slower than techniques for handling modularly stratified programs.

We have incorporated our method into the Glue–Nail database system [10, 11]. An earlier version of the system

employed Ross's method [23]. We have compared Ross's method and our technique. The performance results can be found in Derr *et al.* [11] and Morishita [17]. The results show that, although one can write a modularly stratified program for which our method runs slower than Ross's technique, the converse is also true, depending on the properties of the EDB relations.

The reader might feel that there is no advantage to using our method for modularly stratified programs. It should be remembered that methods for modularly stratified programs suffer from the undecidability of determining whether a given program is modularly stratified for all EDBs [24]. Although there are some sufficient semantic conditions for modular stratification available [23, 24], in general the programmer must guarantee that the given program is modularly stratified in order to get correct answers by using methods for modularly stratified programs. Furthermore, it could be a difficult task for the programmer to ensure this property for complex programs. Our method frees the programmer from this task and can deal with non-modularly stratified cases.

4.3. *Top-Down Procedures*

Besides approaches that adapt the magic-sets transformation, several top-down procedures have been proposed for the well-founded semantics [7, 8, 19, 22]. The magic-sets transformation is a set-oriented approach and is effective for computing all answers in a batch. On the other hand, top-down procedures constitute a tuple-oriented approach that is suitable for obtaining answers step by step. The most well-known top-down procedure for programs with negation is SLDNF-resolution [9, 16], which is commonly used for proving a goal that is a logical consequence of the completion of the program. SLDNF-resolution, however, cannot be directly used for computing the well-founded semantics, because it cannot cope with undefined subgoals. SLS-resolution [19] and global SLS-resolution [22] extend SLDNF-resolution for computing the well-founded semantics. As a theoretical construct, SLS-resolution is impeccable; however, from a practical viewpoint it suffers from the problems of infinite positive recursion, infinite recursion through negation, and redundant derivation of identical subgoals, even for Datalog$^{\neg}$ programs.

To overcome those problems of SLS-resolution, variants of SLS-resolution [7, 8] have been proposed that use the memoing (tabulation) technique [28] to avoid trivial infinite recursion and redundant derivation of identical subgoals. Furthermore, these variants are more goal-oriented than our magic alternating fixpoint technique. They maintain the dependency information between facts, which could be very large sometimes, and try not to perform irrelevant computation. SLG-resolution [8], for example, delays evaluation of a literal that is temporarily assumed to be undefined in the well-founded model. Such suspended literals are handled as conditions on answers in the table, and may later be found to be true or false, which triggers other inferences. If applied to the motivating example in Section 1, SLG-resolution can focus on the nodes a, b_1, b_2, c_1, and c_2 in Fig. 3, while our method scans all nodes in the tree up to the second overestimate. Thus SLG-resolution performs better than our method in the example. In the worst case, however, SLG-resolution has to maintain dependency between a large set of facts, which may reduce the performance. We have not made performance comparisons with the implementation of SLG-resolution on XSB [27]. According to [27], XSB's engine is much faster than Glue–Nail's engine based on our magic alternating fixpoint technique for handling modularly stratified programs and is comparable to Glue–Nail's for stratified programs and non-modularly stratified programs.

ACKNOWLEDGMENTS

I am indebted to Jeff Ullman for numerous discussions that motivated me to pursue this work and for valuable comments on earlier drafts of this paper. Next I thank Shuky Sagiv for discussions on the performance of the method proposed in this paper. I am also grateful to the anonymous referees for insightful comments and suggestions.

REFERENCES

1. K. R. Apt, H. A. Blair, and A. Walker, Towards a theory of declarative knowledge, *in* "Foundations of Deductive Databases and Logic Programming" (J. Minker, Ed.), pp. 89–148, Morgan Kaufmann, Los Altos, CA, 1988.
2. I. Balbin, K. Meenakshi, and K. Ramamohanarao, A query independent method for magic set computation on stratified databases, *in* "Proc. International Conference on Fifth Generation Computer Systems, 1988," pp. 711–718.
3. I. Balbin, I. Port, K. Ramamohanarao, and K. Meenakshi, Efficient bottom-up computation of queries on stratified databases, *J. Logic Programming* **11** (1991), 295–345.
4. F. Bancilhon and R. Ramakrishnan, An amateur's introduction to recursive query-processing strategies, *in* "Proc. of ACM SIGMOD International Conference on Management of Data, 1986," pp. 16–51.
5. F. Bancilhon, D. Maier, Y. Sagiv, and J. D. Ullman, Magic sets and other strange ways to implement logic programs, *in* "Proc. of ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, 1986," pp. 1–15.
6. C. Beeri and R. Ramakrishnan, On the power of magic, *J. Logic Programming* **10** (1991), 255–299.
7. W. Chen and D. S. Warren, A goal-oriented approach to computing the well founded semantics, *in* "Proc. Joint International Conference and Symposium on Logic Programming, 1992."
8. W. Chen and D. S. Warren, Query evaluation under the well founded semantics, *in* "Proc. of ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, 1993," pp. 168–179.
9. K. L. Clark, Negation as failure, *in* "Logic and Databases" (H. Gallaire and J. Minker, Ed.), pp. 293–322, Plenum, New York, 1978.
10. M. A. Derr, S. Morishita, and G. Phipps, Design and implementation of the Glue-Nail database system, *in* "Proc. of ACM SIGMOD International Conference on Management of Data, 1993," pp. 147–156.

11. M. A. Derr, S. Morishita, and G. Phipps, The glue–nail deductive database system: Design, implementation, and evaluation, *VLDB J.* **3**, No. 2 (1994), 123–160.
12. M. Gelfond and V. Lifschitz, The stable model semantics for logic programming, *in* "Proc. of International Conference Symposium on Logic Programming, 1988," pp. 1070–1080.
13. J. M. Kerisit and J. M. Pugin, Efficient query answering on stratified databases, *in* "Proc. of International Conference on Fifth Generation Computer Systems, 1988," pp. 719–726.
14. D. B. Kemp, P. J. Stuckey, and D. Srivastava, Magic sets and bottom-up evaluation of well-founded models, *in* "Proc. of International Logic Programming Symposium, 1991," pp. 337–351.
15. D. B. Kemp, P. J. Stuckey, and D. Srivastava, Query restricted bottom-up evaluation of normal logic programs, *in* "Proc. Joint International Conference and Symposium on Logic Programming, 1992," pp. 288–302.
16. J. W. Lloyd, "Foundations of Logic Programming," Springer-Verlag, New York, 1987.
17. S. Morishita, An alternating fixpoint tailored to magic programs, *in* "Proc. of ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, 1993," pp. 123-134.
18. C. H. Papadimitriou and M. Yannakakis, Tie-breaking semantics and structural totality, *in* "Proc. of ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, 1992," pp. 16–22.
19. T. C. Przymusinski, Every logic program has a natural stratification and iterated least fixed point model, *in* "Proc. of ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, 1989," pp. 11–21.
20. R. Ramakrishnan, D. Srivastava, and S. Sudarshan, Controlling the search in bottom-up evaluation, *in* "Proc. of Joint International Conference and Symposium on Logic Programming, 1992," pp. 273–287.
21. J. Rohmer, R. Lescoeur, and J. M. Kerisit, The Alexander method, a technique for the processing of recursive axioms in deductive database queries, *New Generation Computing* **4** (1986), 522–528.
22. K. A. Ross, A procedural semantics for well-founded negation in logic programs, *J. Logic Programming* **10**, No. 1 (1992), 1–22.
23. K. A. Ross, Modular stratification and magic sets for Datalog programs with negation, *in* "Proc. of ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, 1990," pp. 161–171.
24. K. A. Ross, Modular acyclicity and tail recursion in logic programs, *in* "Proc. of ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, 1991," pp. 92–101.
25. Y. Sagiv, Is there anything better than magic?, *in* "Proc. of International Logic Programming Symposium, 1990," pp. 235–254.
26. H. Seki, On the power of Alexander templates, *in* "Proc. of ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems, 1989," pp. 150–159.
27. K. Sagonas, T. Swift, and D. S. Warren, XSB as an efficient deductive database engine, *in* "Proc. ACM SIGMOD International Conference on Management of Data, 1994," pp. 442–453.
28. H. Tamaki and T. Sato, OLD resolution with tabulation, *in* "Proc. of International Conference on Logic Programming, 1986," pp. 84–98.
29. J. D. Ullman, "Principles of Database and Knowledge-Base Systems," Vol. 2, Computer Science Press, New York, 1989.
30. A. Van Gelder, K. A. Ross, and J. S. Schlipf, The well-founded semantics for general logic programs, *J. Assoc. Comput. Mach.* **38**, No. 3 (1991), 620–650.
31. A. Van Gelder, The alternating fixpoint of logic programs with negation, *J. Comput. System Sci.* **47** (1933), 185–221.